Comput Visual Sci (2015) 17:167–183
DOI 10.1007/s00791-015-0256-9

Architecture for modeling and simulation of technical systems along their lifecycle

Tim Schenk[1] · **Albert B. Gilg**[1] · **Monika Mühlbauer**[1] · **Roland Rosen**[1] · **Jan C. Wehrstedt**[1]

Received: 8 July 2015 / Accepted: 16 December 2015 / Published online: 27 January 2016

Abstract Modeling and simulation is an established scientific and industrial method to support engineers in their work in all lifecycle phases—from first concepts or tender to operation and service—of a technical system. Due to the fact of increasing complexity of such systems, e.g. plants, cyber-physical systems and infrastructures, system simulation is rapidly gaining impact. In this paper, a simulation architecture is presented and discussed on three different industrial applications, which offers a client–server concept to master the challenges of a lifecycle spanning simulation framework. Looking ahead, open software concepts for modeling, simulation and optimization will be required to cover new co-simulation techniques and to realize distributed, for example web-based simulation environments and tools.

1 Introduction

Purpose and objects of computer-based modeling and simulation have evolved significantly since their first applications mid of last century. It is now an established scientific and industrial analytics and design technology with strong focus and strength to analyze in particular specific physical product features [1,2]. Numerous powerful tool implementations are available for use as single stand alone tools and increasingly integrated into industrial engineering workflows. But the simulation focus continues to address larger systems, not just components or small products. In the industrial context this involves tasks of such large and increasingly complex systems over their whole lifecycle—from first tender to operations and service. Examples range from decision making during bidding and conceptual design, detailed engineering, testing and commissioning, as well as optimized runtime operations and service [3]. Lumped, so-called 0D simulation, and discretized 1D simulation are main technologies to address these tasks[1] and the presented architectural approach tries to get a grip on them.

The considered fields of application include large process and production plants, power stations and grids, communication networks, different kinds of machines and vehicles as well as interconnected heterogeneous infrastructure systems. All these systems can be characterized by network-type structures composed of a multitude of typically heterogenous interconnected components. Another important key characteristic of such industrial systems are their automation systems for process control by open- and/or closed-loop mechanisms on different levels of detail.

Several commercially available tools, like Matlab Simulink [4], AMESim [5] or Modelica-based Dymola [6], address the simulation of such technical systems. They focus on certain well-confined engineering tasks in specific lifecycle phases—mostly detailed engineering—and often trade in computational efficiency (speed and system sizes) against restrictions in modeling capabilities and system characteristics. Recently, such problem centric modeling and simulation tools become challenged by further demands. First of all, soaring effort and cost of modeling and tool development are limiting factors to further progress. Hence, the abundance of available models, model libraries and tool implementations seems to be an attractive resource and value to reuse concepts.

Communicated by Gabriel Wittum.

✉ Tim Schenk
tim.schenk@siemens.com

[1] Siemens AG, Otto-Hahn-Ring 6, 81739 Munich, Germany

[1] 2D and 3D simulations are additionally used especially for detailed design and engineering challenges.

But heterogeneity and missing interoperability are critical obstacles to pursue such approaches. Secondly, the continuing progress of computing hardware and network band width opens opportunities to break up complexity bottlenecks. Recent trends, like multi–core hardware architectures, distributed and cloud computing demand for new systematic program structures and data flows but are laborious and tricky to be implemented and exploited [7]. Modeling and simulation architectures need to efficiently support and take advantage of these computing resources and paradigm evolutions. Thirdly, the involvement of different disciplines and domains in the system development process further increases system heterogeneity and poses additional challenges regarding the interplay of multiple disciplines, such as mechanics, electronics, software and communications, during system evaluations [8,9]. And the system viewpoint is even further expanding to a lifecycle view, which allows for increasing quality and efficiency of system engineering and operation, e.g. system design for reduced lifecycle cost.

The simulation architecture described in this article avoids the discussed restrictions of existing tools and faces the upcoming challenges as it allows utilization of a single simulation environment throughout all lifecycle phases and still offers the appropriate application support for each task at hand by deployment of customized simulation solutions.

The following section introduces three industrial examples of infrastructure and plant systems—a pumping station for drinking water, the heat recovery module of a power plant and a sewage infrastructure network—which exemplify and illustrate actual simulation issues of real projects. The section also introduces a further key factor to tackle the complexity of modeling and simulation tasks by differentiating between the views of engineers and tool developers.

The subsequent Sect. 3 clusters features of the examples in Sect. 2 and defines characteristic use cases that cover a typical range of tasks during all lifecycle phases. The use cases reveal the technical requirements, which have to be addressed by the architectural concept for a modeling and simulation framework.

The modeling and simulation framework CoSMOS (Complex Systems Modeling, Simulation and Optimization)[2] is introduced and outlined in Sect. 4. Its concept as well as the general modular architecture is presented. Two main aspects of CoSMOS—its focus on a client server architecture and its embedded simulator—are discussed in detail.

Section 5 exemplifies the applicability of CoSMOS and its specific features along the three industrial examples and their individual simulation tasks introduced in Sect. 2.

Together with a short summary, the final section delivers an outlook on some specific issues still open to be solved as well as future challenges.

[2] In house simulation framework of Siemens Corporate Technology.

2 Complexity of system simulation in industrial usage

Industrial simulation is increasingly challenged by system complexity since large systems have to be captured in their entirety and in many details. Moreover, simulation tool users and in particular tool developers have different views, i.e. goals and expectations on necessary properties of a simulation tool. In the following, three examples of industrial applications as well as two characteristic views of different stakeholders are presented to familiarize the reader with the issues faced by industrial system simulations.

2.1 Industrial examples

This selection covers a representative variety of different applications covering different phases of a system's lifecycle.

2.1.1 Pumping station

Water pumping stations enable water transport between remote locations. Simulations are used throughout all lifecycle phases, from early concept phases to the commissioning phase, plant operations and beyond [10].

The pumping station presented in this example shall transport water from a salt water desalination station through a twin pipeline of 20 kilometer length to an intermediate reservoir situated 500 meters above sea level. From there, the water flows to a receiving and a tap-off station driven by gravity, see Fig. 1. The system comprises several pipelines, tanks with level sensors, valves and pumps to be controlled.

In the contract bidding and conceptual phase, several basic design alternatives are evaluated with respect to function, performance and cost of required components with the help of simulations. In general several parallel booster pumps and the same number of parallel main pumps are used. Design alternatives to reduce pressure in the pipeline have to be evaluated, e.g. a shift of the main pumps to a distant location. The tradeoff is to construct an additional pumping station at a location halfway towards the reservoir with additional cost and effort, e.g. caused by accessibility problems. In such early lifecycle phases only draft models of the components and the system layout are available.

During the engineering phase, modeling and simulation can support fundamental decisions on specific control and operation concepts. For example, the so-called bypass concept, see Fig. 2, can be configured rather easily, even without detailed engineering expertise. It allows for a partial reflow of the pumped water in order to avoid pump destruction due to cavitation. Improper operation may even lead to pressure shocks with high risk for damages to the pipelines.

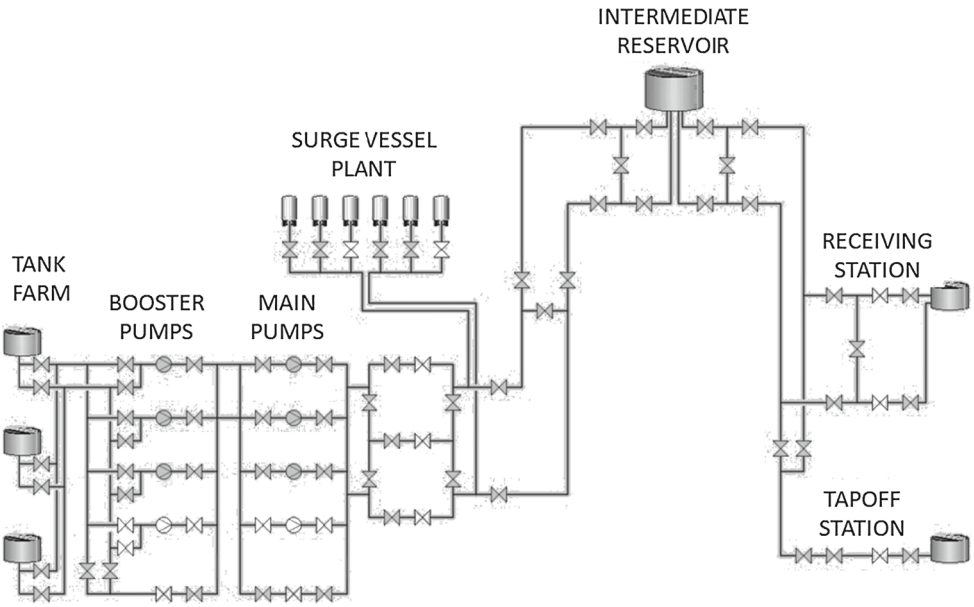

Fig. 1 Schematic of a pumping station and the related water network

Even before commissioning and later during the operation phase of the pumping station, system simulation models are used to train the plant operators in a virtual environment. Optimized plant operations are made possible through model-based assist systems. They provide prognoses of the future system behavior, hence giving the opportunity to predict and include external influences, like supply changes from the desalination station, and develop optimal control strategies. Furthermore, malfunctions of the pumping station can be detected early by online diagnosis and comparing actual real operation data with simulated data. Lastly, refurbishment and plant extensions can be planned efficiently using a virtual plant model before their implementation.

Although all the above described tasks evaluate the same water network, the underlying utilized models and algorithms are very task-specific in detail. To guarantee an efficient continuous engineering, the models and results have to be designed and managed consistently—horizontally along all phases and vertically in levels of detail.

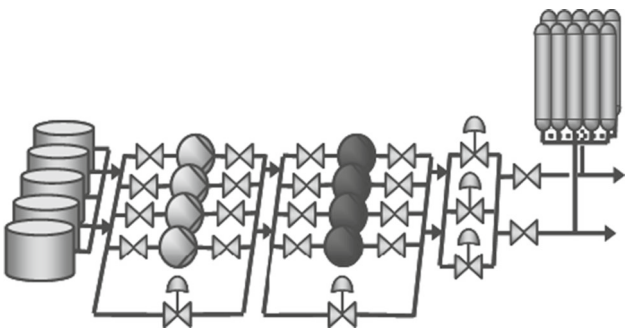

Fig. 2 Bypass concept for a pumping station

2.1.2 Heat recovery steam generators (HRSG)

This example focuses on detailed engineering of heat recovery steam generators and the benefits of simulations.

In combined cycle power plants the enthalpy flow of the hot exhaust gases is exploited in a secondary water / steam cycle to increase overall plant efficiency and power output. A key system is the heat exchanger between the two fluid cycles, the so-called Heat Recovery Steam Generator (HRSG). For certain HRSG technologies, like the once-through design [11], see Fig. 3, it is essential to analyze the dynamic flow

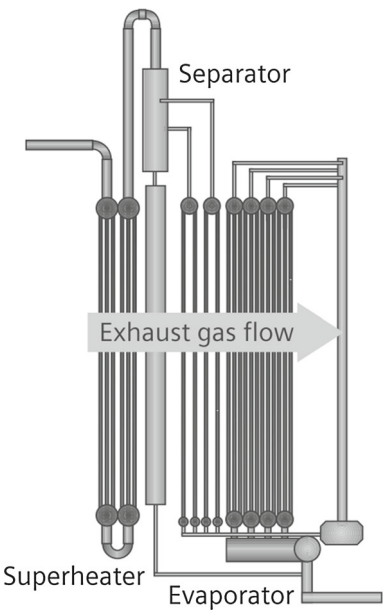

Fig. 3 Once-through heat recovery steam generator

stability in the design phase in order to ensure a controlled state at any time during plant operation. For instance, the design of valves might need to be adapted to avoid oscillations in pressure and mass flow with high risk of pipe bursts and leakages.

Dynamic analysis of the flow inside such an HRSG is quite tricky: Discretization of the HRSG system leads to a large number of equations with several thousand degrees of freedom and it contains numerous design parameters such as number of pipes, their positioning, lengths and diameters. Besides, initial values for the flows have to be provided for calculation. Hence, a manual setup of the model is very time-consuming and error-prone. This is overcome by an automatic mapping from previous steady-state simulations, which were performed to fix a primary design of the HRSG.

Finally, beyond the HRSG design tasks, it is of interest to boiler manufacturers and also to OEMs to include the boiler into the overall plant system model as well. For control design and optimization, these plant models are used for early validation and to define best operation strategies (such as quick ramp-ups to catch attractive power price windows). To describe the overall plant system, both fluid cycles (gas as well as water / steam) together with the control system have to be modeled in addition to the HRSG. Third party tools may provide suitable libraries for additional components like gas and steam turbines, condensers, pumps or control blocks. Thus it is attractive to be able to connect several submodels (HRSG and "periphery") to provide an overall system simulation model. Model integration techniques that allow formulation as a single overall system or co-simulation methods using independent tools become key issues of industrial system simulation.

2.1.3 Sewage network

The third industrial example illustrates operational support of sewage networks with respect to resource efficiency and environmental restrictions enabled by simulation forecasts and optimization algorithms [12,13].

Sewage networks, illustrated in Fig. 4, in cities nowadays do not include much automation and control infrastructure. They are mostly run by small local controllers and experienced operators. Main obstacles to include more optimal control equipment are: very small economic benefit for the operator companies (mostly public services), despite of public and environmental benefits, the uncertainties to predict system inflow profiles from rain and sewage producers and difficulties to understand handling alternatives and their long time impact in such a sewage network.

However, the increasing occurrence of more and heavier downpours, the fact of low city budgets for infrastructure investments and the increasing environmental awareness of

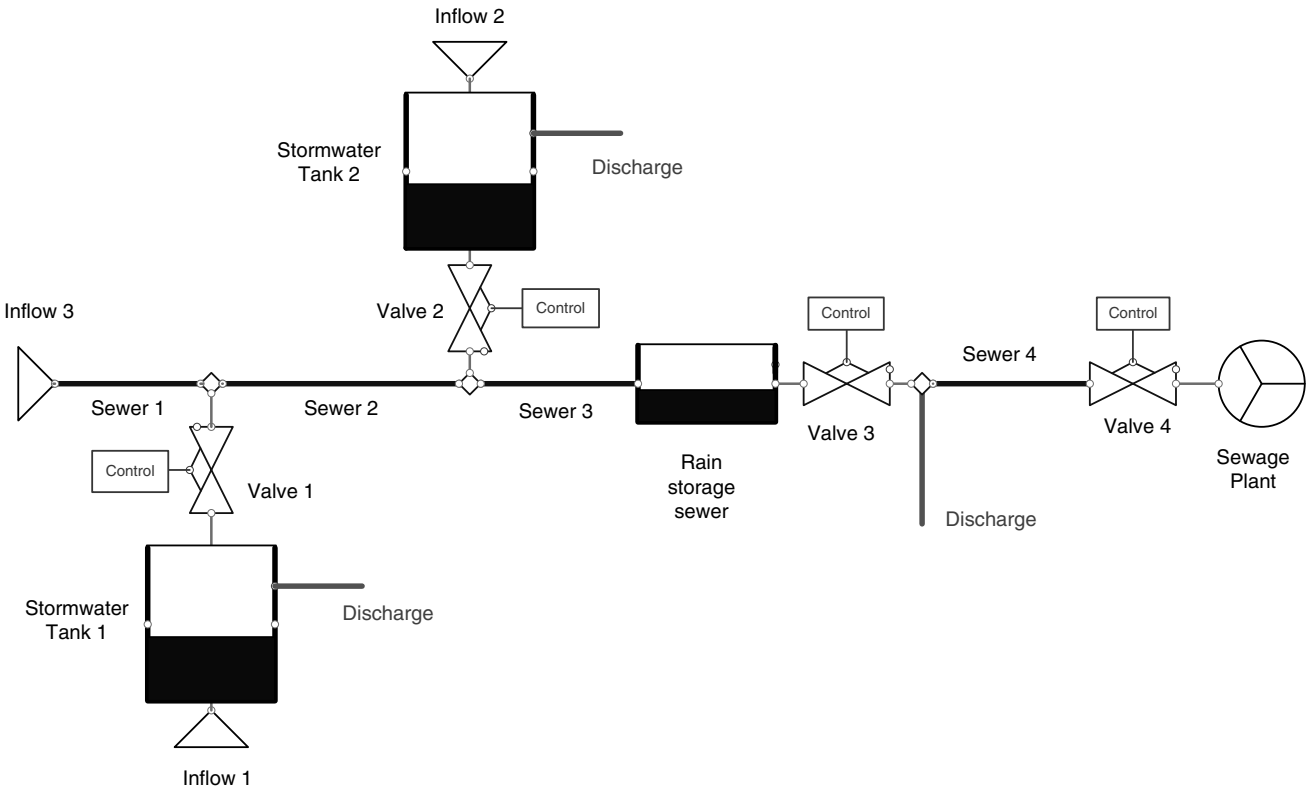

Fig. 4 Functional scheme of a small sewage network

the public call for better operation controls. Cheaper sensor technology and availability of wireless communication infrastructure in combination with efficient simulation models are cornerstones to successfully develop and provide IT-based operational support. This enables solutions for different tasks: Network monitoring of actual and prediction of future network states; forecasting of consequences for different selected control strategies; automatically generated suggestions for changes of valve and pump settings for different operation points like for example rain events. Even a 24/7 fully automated, optimal control of a sewer network may become possible.

Some of these tasks can use offline simulation runs, e.g. concept studies helping in investment decisions or in reviewing the necessity of control decisions during past events. This simulation requires a 1D simulation of wastewater flow, pressurized and open channel flow, through the network. But most of the tasks need online simulation capability in the control center during network operation and will be directly connected to the Supervisory Control and Data Acquisition (SCADA) system to access actual sensor data (e.g. flow values, filling levels, valve positions, rain data).

In particular, tasks performed during operation should be able to run in parallel or in a specified sequential order. Each of these tasks may be configured differently regarding when and how it is executed. For example, the network monitoring shall run every 15 minutes, but a forecast or an optimization for an upcoming downpour shall be triggered by the operator—without stopping the monitoring.

2.2 "Bi-view" engineer vs. IT approach

Besides the complexity of the applications themselves, system simulation environments have to fulfill very different requirements of their stakeholders. There are two main groups of stakeholders.

First of all there are users, i.e. engineers, who have to perform specific tasks in dedicated system lifecycle phases. They require one or if not avoidable several simulation tools that hopefully support perfectly. Considering for example the setup of a component-based simulation model, an engineer's work typically comprises the instantiation of all components, stemming eventually from different component libraries and defining their interconnections. Furthermore, the setting of parameters, like geometric dimensions, and of initial component states and boundary conditions, such as the initial filling level of tanks, needs to be performed. After specifying simulation run configurations, like simulated time intervals and a preferably small set of solver settings, simulation runs are executed and results will be evaluated during and/or after the simulation run via graphical viewers. The engineer requests (semi-)automatic support by the tool for his tasks whenever possible to reduce the work load. This ranges from automatic model generation, help in setting a most suitable simulation configuration for specific tasks up to scenario management and result interpretation.

This engineer's focus is not on underlying tool concepts such as algorithms and implementation, software service and upgrade concepts. These are in focus of the other group of stakeholders, the tool vendors and developers and their IT centered viewpoint. Their target is to cover as much customer demand as necessary while keeping cost of implementation and service as low as possible. One key goal is to enable all the functionality for each type of simulation task in a user-friendly way. This comprises, among other things, the development and supply of component libraries in various levels of detail, as well as advanced solving algorithms for large systems of equations with time continuous as well as time discrete variables. Additionally, further demands regarding the time management can arise, like real-time requirements.

These demands resulted in a vast landscape of task and application specific simulations tools. Main challenges are reduction of modeling cost by reuse of existing models and interoperability of dedicated tools. Promising techniques like co-simulation via flexible interface descriptions are becoming established (e.g. the Functional Mockup Interface [14]).

3 Use cases and their technical requirements

Reviewing the industrial examples and both viewpoints from Sect. 2, this section defines characteristic use cases and the consequent technical requirements for the modeling and simulation architecture.

3.1 Use cases

The following three use cases address the main phases of a system lifecycle with quite different simulation objectives. Many more use cases could be defined during designing, developing or operating technical systems, products and plants. Yet, this selection is quite generic so that most other use cases should be minor modifications or a combination of the three ones presented here.

3.1.1 Decision making in early planning and conceptual design

Developing a new technical system starts with a conceptual planning phase. The goal is to create the layout of the system, outline its main boundaries, content and parameters. This phase is characterized by abstract or at most half-detailed technical system specifications. Most of the tasks are performed by a small team of system designers. Expertise and feedback from customers and domain experts is collected

selectively. Design alternatives with respect to the system requirement specification and other relevant project factors (feasibility, cost, time, etc.) are developed, evaluated and compared.

Modeling and simulation approaches supporting the decision making process need to provide an administrable experiment management (e.g., for altering evaluating and comparing design parameters) including a transparent display of the relevant differences of the evaluated system alternatives. Furthermore, the modeling and simulation application has to be easy-to-use for non-simulation experts and executable on "standard" hardware such as laptop and desktop computers. The simulation has to perform abstract multi-domain evaluations and behavior models of the system. Its results have to be processed further and therefore need to be embedded into the engineering workflow (concept transfer).

3.1.2 Detailed engineering and commissioning

After specifying the system design, it is handed over to the domain engineers. They perform detailed development for all parts of the system. This is supported by specialized expert software in many cases. Objective of this phase is to specify each single piece of the system to guarantee its functionality, to meet the requirement specification and remain within the limits and boundaries governing the conceptual design. For today's complex systems, these tasks often enforce a joint evaluation of different engineering domains including technical process and automation. Many of the individual engineering processes run concurrently, thus demanding a recurring evaluation of the system along detailed development steps.

Because detailed engineering is mainly supported by several specialized, in most cases single domain dependent, commercial software and expert simulation tools, it is necessary to support the emerging concurrent development by modeling and simulation techniques that enable the coupling of different simulation models, algorithms and tools. Sometimes different levels of detail with different time restrictions (up to realtime capability) of the system have to be simulated simultaneously. Simulations have to be performed in different environment settings, e.g. hardware-in-the-loop (HIL), software-in-the-loop (SIL), standalone software. System parameters, data and results have to be transferred and exchanged between different engineering tools. Validation and test as well as virtual commissioning are the most known simulation tasks at the end of engineering.

3.1.3 Runtime operation

When the system has been developed and realized, modeling and simulation support moves nearly entirely to runtime relevant tasks. They enable operator training, production as well as operational assistance and support of diagnosis and service. Goal is to guarantee optimal operation of the system with respect to time, quality and cost. Users of such applications are often no simulation experts but production or control center operators. These applications exploit online sensor data, preprocess them, evaluate calculations bound to real-time and pass results, mostly in abstract or aggregated form, to further post processing applications.

Thus the modeling and simulation approach has to enable consistent system specification and model import from the engineering phase. Algorithms must have real-time capability (or sometimes much faster) and run integrated in a multi-application, multi-data IT environment. Different levels of details of data and models need to be combined and aggregated [15]. Easy handling of otherwise tricky simulation tasks (e.g. monitoring, forecasting, system state estimation, optimization, etc.) is necessary.

3.2 Technical requirements

Efficient handling of the before mentioned use cases demands an architectural concept for tool, model and data integration and management, which fulfills several technical requirements.

3.2.1 Different calculation executions

Examples of different calculations are steady-state and dynamic simulations, co-simulation (e.g. of technical process and automation) and numerical optimization. This requires an architectural concept allowing specific configurations for each calculation and multiple calculations to be performed sequentially or in parallel. During execution of a single calculation multiple interacting applications have to be orchestrated. The architecture concept must be open for modular expansion to configure new calculation types necessary for further applications.

3.2.2 Life-cycle consistent, exchangeable models

The deployed simulation models, that are behavior models and system topologies, evolve along the system lifecycle. Starting from coarse grain, often just steady-state models in early phases, they grow in the engineering phase to very sophisticated models including dynamic effects and may even become modified to meet special requirements of commissioning and operation phases, where real-time capability becomes crucial. Along the lifecycle, the usage and transformation from one model to the next has to be consistent, transparent and easy to realize.

Therefore the architecture has to support a traceable model administration for behavior implementation and system layout. This includes at least a provision of interfaces and

meta-models to allow an easy switching between the different levels of detail. In addition, mechanisms for picking a suitable model behavior and take over, aggregate and break down the required data and parameters should be provided.

3.2.3 Adaptable user interfaces

Each phase requires different user frontends for the general control of the simulation application, like starting, stopping, scheduling, configuring, etc. During concept and engineering phase using desktop or laptop computers, a specific graphical user interface or a commercial application on a standard operating system, like Windows, is needed. During late lifecycle phases simulation is integrated into a real-time, online environment without a specific simulation user interface and is triggered as an internal software module by e.g. the command and control center or HIL/SIL applications.

Hence it is required to have generic control interfaces available to trigger the calculation execution by various front ends and software environments. As each application needs a graphical user frontend for the engineering of the system topology and result exploration, it shall be easy to build and include those proprietary GUIs. Such GUIs typically support configuration of controls, topology definition, parameterization and result exploration.

3.2.4 Integration into multi-application and multi-data environment

Most use cases involve calculations of multiple applications that interact with each other during runtime. The focus of the interactions is provision and exchange of dynamic variables of different applications, like setting a motor speed, opening a valve or injecting fault states in operator training. Depending on the use cases this data can be exchanged between different simulators (realizing co-simulations), a real SCADA system and a simulator (during operational usage) or even some proprietary applications like a failure injector. The dependency between the applications and their exchanged variables can be different—some have to compute in parallel (like in co-simulations), some sequentially (e.g. when overtaking sensor data).

It has to be easy to integrate any participating commercial or proprietary application into the simulation architecture and schedule and configure its data exchange. The treatment of these data and variables has to be implemented efficiently as there can be loads of data that have to be exchanged.

3.2.5 Hybrid simulation models and algorithms

Since simulation is used in different development phases and deployed to solve different tasks, focus and goal of the simulation vary. Often multiple domains and disciplines are simulated together. For every specific task—usage, purpose and involved disciplines—behavior models and mathematical algorithms have to be implemented task-oriented.

Often, there is no commercial simulator available with models fulfilling the requirements—especially in early and operational phases or also if multi-disciplinary models get involved, e.g. for simulation of the technical process and its automation. Model behavior can be time discrete, event discrete or time continuous—or even a combination in a hybrid model. Algorithms have to be available that can evaluate the dynamic behavior of all these three types intertwined in one simulation.

4 CoSMOS philosophy: integration into a seamless engineering workflow by coupling of models and simulators with engineering and real time tools

4.1 Focus and goal of CoSMOS

Considering the aspects and requirements from Sects. 1 to 3, Siemens Corporate Technology developed a modeling and simulation framework with the acronym CoSMOS ("Complex Systems Modeling, Optimization and Simulation"). The framework is utilized to develop simulation-based solutions in various business units of the Siemens company. Completely different technical systems are addressed and a variety of tasks in different phases of their lifecycle are covered. The usage ranges from pilot and demonstrator projects to in-house engineering support tools up to Siemens portfolio solutions. Examples can be found in Sect. 5.

CoSMOS aims to evaluate the kind of technical systems introduced in Sects. 1 and 2, ranging from infrastructure networks to plants of any kind to complex machines. It is designed to be individually customizable to meet all the required technical aspects and tasks that are described in Sect. 3.

Therefore CoSMOS is able to orchestrate the interplay[3] of different applications, calculation and simulation modules. Furthermore, the framework allows a continuous usage of the same simulation tool environment over all lifecycle phases of a system and provides integrated, deployable, easy-to-use solutions in particular for semi-expert users like domain engineers, operators or sales staff.

A CoSMOS developer can customize the modeling, simulation and optimization tasks and wrap and hide the complexity of the underlying algorithms, mathematics and physics. This includes the system topology, behavior models, numerical algorithms, the in- and output data processing,

[3] This requires remote control of applications: time control and data exchange.

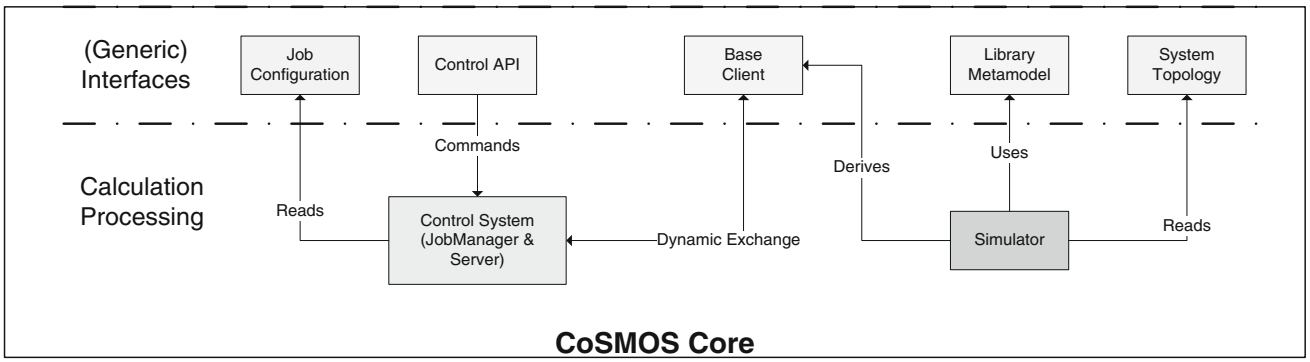

Fig. 5 CoSMOS core elements

pre-configurations and the interplay of various participating modules and applications.

4.2 Architecture

Functionality in the CoSMOS core can be divided into two main parts. On the one hand, there are modules that are responsible for the processing of the calculation and on the other hand there exist generic interfaces for modular software setups enabling a wide range of solution specific customization (see Fig. 5).

Calculation processing is performed by the main CoSMOS *control system* module and an available proprietary CoSMOS *simulator* submodule.

The *control system* is the essential main module of CoSMOS. It is responsible for the management of different jobs (=calculations), the correct preparation and setup of each of the jobs and the sequence control of a single job including all its involved parties called clients in CoSMOS language and their intercommunication. A more detailed insight is given in Sect. 4.3. The term job corresponds in the CoSMOS environment to a simulation/calculation run (see Sect. 4.3.3).

The *CoSMOS simulator* is a proprietary fully-fledged dynamic simulator containing an own internal sequence control for an integrated discrete-eventcontinuous evaluation of the instantiated simulation models (see Sect. 4.4). The CoSMOS simulator is optional and not a required participant (client) in a CoSMOS job execution.

To be as flexible and modular as possible, CoSMOS provides numerous generic interfaces that can be used and implemented. To begin with there are extensible, xml-based configurations for the job and for the system topology. The job configuration defines all general settings of a single simulation job and is processed by the job management in the main control system. Parts of this configuration

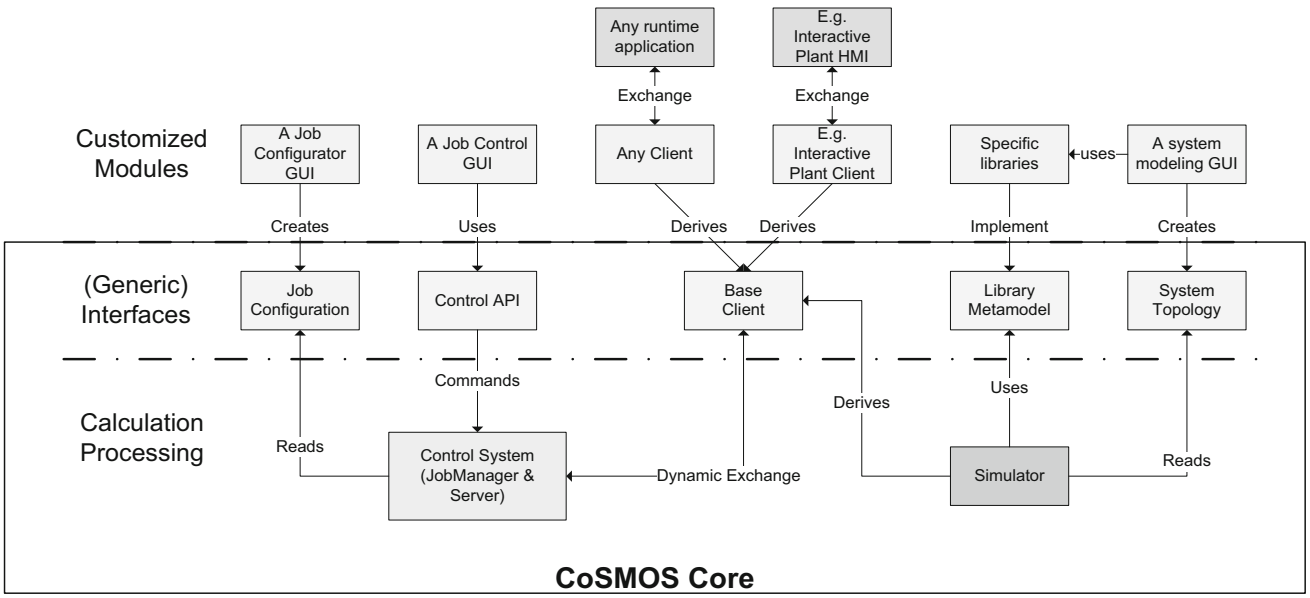

Fig. 6 Exemplification of a generic CoSMOS solution

are the time control settings, the specific configuration of each participating client and the management of results. The system topology is stored separately in another configuration file following the component-based principles stated in Sect. 4.1 consisting of components, parameters and connections. This configuration can contain several differing versions of the same system depending on the purpose of further processing. For example, there is at least a version for the general layout containing components and their graphical representation. Each participant (client) that processes proprietary system topology information can add a specific version containing the client specific information. For example, the proprietary CoSMOS simulator adds a version containing simulation relevant information, like interconnection types and values of parameters.

Besides these two configurations, CoSMOS provides base classes and meta models that can be implemented to extend and add functionality. There is a base client to be implemented by any application that wants to participate in a simulation run and therefore becomes a client (see Sect. 4.3.1). To be able to manage various jobs and control the execution of each single job, there is an API control interface that provides functionality for creating, loading, starting, stopping, etc. of jobs. Providing the CoSMOS simulator with own libraries containing behavior models can be done by implementing the library meta model (see Sect. 4.4), as it is available in most commercial simulators, like the ones stated in Sect. 4.1.

For exemplification, a usage and implementation in a classical programming language like C++ or C# of all the interfaces is shown in Fig. 6 by a generic customization. This way a complete application serving a specific purpose can be built—also called CoSMOS solution.

The presented architecture supports implicitly a generic engineering workflow. Depending on the solution, specific modules required for modeling, configuration, control and the runtime participants (clients) can be customized, integrated and configured.

The CoSMOS control system can be operated with different specific GUIs or even completely hidden behind an existing HMI system like PCS 7 OS [16] for use in a control center and automatically executed by it. Stored job configurations can be reused and extended, e.g. predefined offline and loaded during the operation. And participating applications can be flexibly added and removed by loading and unloading their clients, e.g. to communicate between a process simulator and a emulated programmable logic control (PLC) in a Software-in-the-loop test environment or to write simulation data to MS Office applications when doing concept design studies. System topologies that were created in a former development phase can easily be reused in later evaluation by e.g. exchanging the library with a more detailed one.

4.3 Control system

4.3.1 Client–server concept

Looking at a single job execution, the required flexibility in terms of number and kinds of interacting runtime participants is guaranteed in CoSMOS by a so-called client–server architecture (other publications refer to such an approach as master-slave architecture). The approach is similar to some commercial tools like the co-simulation environments TISC [17] or ICOS [18] with a broad and generic scope.

Each single job performs a dynamic calculation that progresses forward in simulation time. Various applications can participate in such an execution via their own client and deliver as well as receive dynamic data (variables) to and from the server. The central server takes care of the sequence control, the triggering of the clients and the correct and efficient data exchange between them (Fig. 7).

Existing commercial or proprietary applications can take part in a CoSMOS job execution just by implementing their specific client interface. The client is derived from a provided base client class which manages already all server-client communication aspects. When a specific solution configuration requires the participation of an application, the client is loaded through dll-runtime loading and its interface routines are automatically triggered by the server.

During runtime, the clients mainly exchange dynamic variables via the server but sometimes also influence the server's sequence control itself (necessary particularly in co-simulations).

To guarantee the correct data exchange, prior to an execution the matching of exchanged variables has to take place. All participating clients define a priori the variables they are sending (writing) to the server, of course exclusively. After-

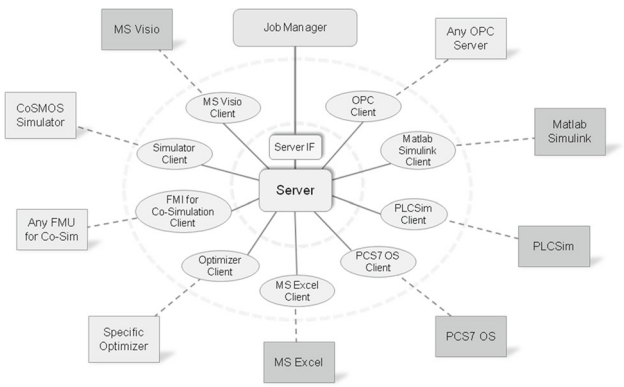

Fig. 7 Client–server architecture

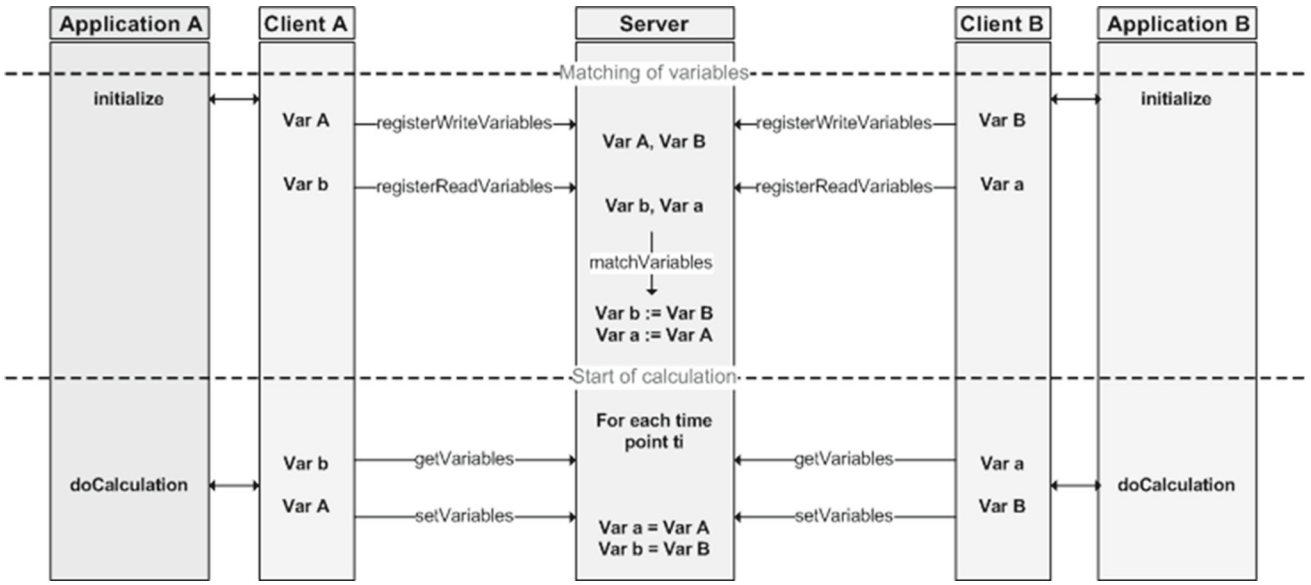

Fig. 8 Data exchange from and to clients

wards, the variables each client wants to obtain (read) are linked to one variable in the list of all available variables—either automatically or defined manually by the user. There are available automatic matching routines like pure name matching, but individual matchings can be implemented, added and used as well. During a job execution, the server triggers each client at specific time points depending on the client configuration (read variables, do calculations, write variables, see Fig. 8).

The trigger time points can differ from client to client, depending on the simulation task and the purpose of each client. When more than one client has to be called at a trigger point, the clients are per default evaluated in parallel. E.g. two simulator clients, running in a co-simulation, would probably have the same trigger points, while a client that updates only a visualization in e.g. MS Visio or PCS 7 OS (Siemens control station HMI) can have longer time steps between its trigger points than the simulators. Usually, trigger points are defined before the actual job execution, but detection of certain events during a simulation can lead to additional trigger points which will be added on the fly.

4.3.2 Scheduling of clients

To execute dynamic simulations, the server control progresses forward in time and calls the participating clients at their trigger time points as stated above. Naturally, all participating clients can simulate dynamic behavior with the same clock—or in other words can integrate in each step from the actual trigger time point to the next. But often enough, they do have an own, different internal clock (e.g. performing a forecast) or just perform a proprietary sort of data processing or similar. Examples for that are the visualization of simulated variables in an individual graphical user front-end, preprocessing of data for the further usage in the simulator and even optimizations and the calculation of KPIs at a specific point in time. As the architecture of CoSMOS is based on the modular client–server concept, all those functionalities are not implemented as add-on to a simulator itself, but are realized as capabilities of the clients that exchange their data via the server. This way they can be reused in all kinds of CoSMOS solutions with all kinds of simulators.

As stated above, clients with a common trigger time point are called by default in parallel to do their calculation. If required, it is also possible to schedule the calls of "stationary" and "dynamic" clients at trigger time points in the correct order. E.g., when for a specific task the simulator is in need of data that has to be preprocessed in each step, the client performing the preprocessing is scheduled before the simulator. This way the data required from the simulator is already present in the server. Another example is the need for visualization during a simulation execution. The visualization client naturally should show the actual values of the variables. Therefore it has to be scheduled after the simulator to get the newly calculated values (see Fig. 9).

4.3.3 Job concept

So far the explained concepts deal with the execution of single simulation runs that can be executed immediately and run as fast as possible. As in the industrial examples and use cases already stated, it can also be necessary to schedule simulation runs, couple them to real-time or even perform multiple simulation executions in parallel or in specific orders—primarily

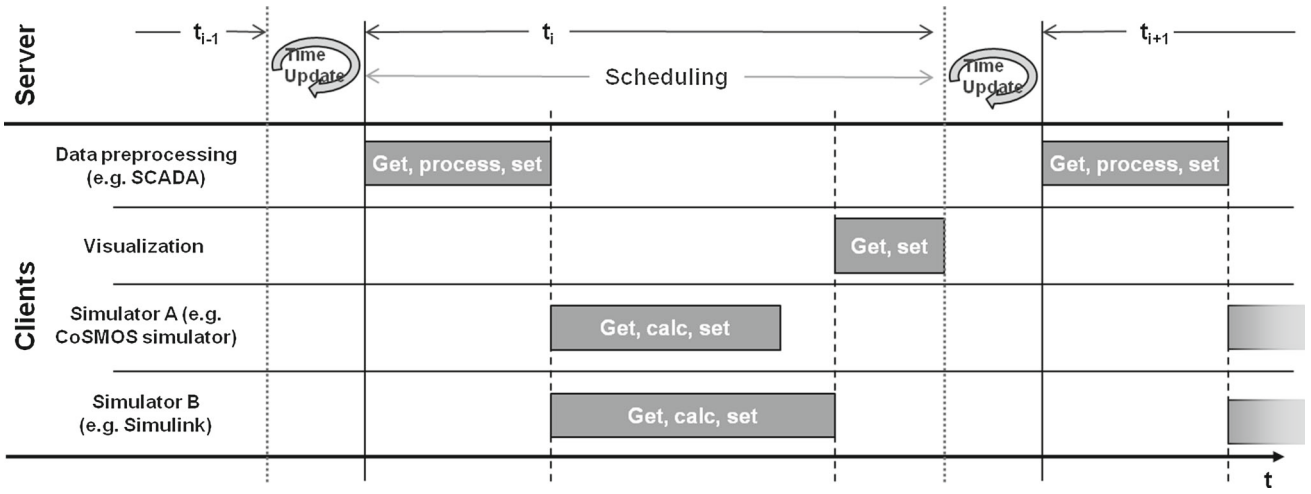

Fig. 9 Example of client scheduling

these requirements appear in environments like experiment and test management and operational support. In CoSMOS, this possibility has been implemented with the job concept.

The concept defines each simulation run as a single job. One job contains a complete client–server setup as described in Sects. 4.3.1 and 4.3.2. The job concept adds to the configuration of a single run—consisting of the setup and configurations of clients and some basic run settings—an extensive time control configuration. Each job can be scheduled with a start time in real time and synchronized with real-time at specified synchronization time points. Moreover, the job can relate its time behavior to real-time labels (virtual real-time).

Complex setups like real-time clocked co-simulations can be performed (see Fig. 10). Simulation steps can be executed periodically in the real-time environment of an operational support system.

To correctly handle jobs, a job manager is implemented in CoSMOS. Apart from managing only a single job, the job manager can be fed with multiple jobs and configured how to deal with them. If the number of processing units allows it, jobs that overlap in their execution time interval will be executed in parallel on different processors. If no free processor is available or it is not intended to execute jobs in parallel, the jobs are executed due to their predefined priorities. A job of minor priority then has to be stopped for later resumption and a job of higher priority is executed first instead.

4.4 Simulator

Besides the possibility to integrate any (commercial) simulator via the client concept, a proprietary simulator is also available in CoSMOS. As every other participant it is implemented as a CoSMOS client and exchanges dynamic variables with the server. This simulator performs efficient dynamic simulations.

The model concept followed by the simulator is component-based as described in Sect. 4.1. This means that the behavior of the whole system results out of the behavior of the instantiation of the types of components and their interconnection between each other.

Thus the CoSMOS simulator builds its internal system simulation model by processing its respective version of the system topology configuration (Sect. 4.2) and subsequently instantiating and parameterizing every component and connection of the topology from the available model libraries. At last the ports of the component instances are connected appropriately to ensure the physical, logical and signal flows between the components.

The sequence control of the CoSMOS simulator copes with event-discrete, time-discrete and continuous models at the same time and furthermore enables differential-algebraic equations (DAEs) in the continuous modeling, which are solved in one complete equation system altogether by special, time efficient DAE-solvers for networked systems in a mathematical sense [19]. This allows also the solution of discretized partial differential equations. The sequence control follows a natural hierarchy, based on the principle of the automation pyramid (see Fig. 11): The time discrete evaluation interrupts the continuous integration for exchange of information, whereas events trigger a specific evaluation in the time discrete and subsequently in the continuous model.

Technological processes are mostly modeled continuously being a synonym for a description by (partial) differential algebraic equation systems, consisting of equations and independent variables. These are evaluated on a continuous time level and require special numerical solvers. Every component modeling continuous behavior can define independent variables (DoFs—degrees of freedom) and implement the

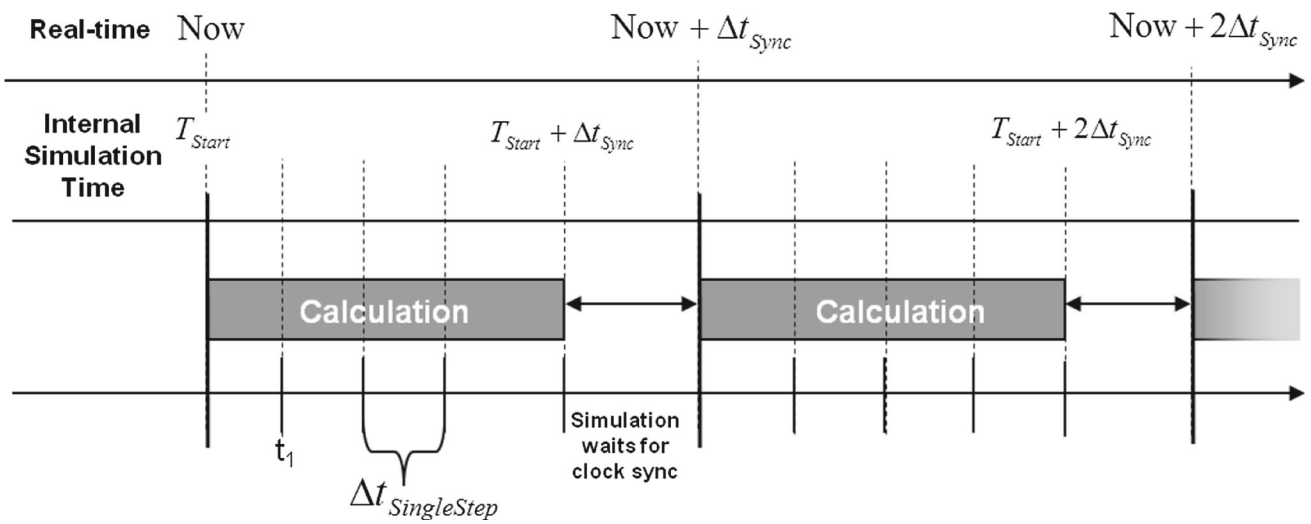

Fig. 10 Real-time synchronized simulation

equations (algebraic and/or differential) in the function calls provided by the meta-model. The CoSMOS simulator collects all DoFs and equations of all instantiated components to build one system of equations and a vector of DoFs. Figure 12 illustrates this approach with a pipe connecting to tanks. The numerical integration of the DoFs describes the behavior of the continuous part of the modeled system. As special types of equations require special numerical solvers, CoSMOS offers the possibility to integrate different numerical solver.

In time and event discrete modeling, usually certain states are defined, which are evaluated either every fixed discrete sample rate or when events occur. Thus automation and control logic behavior is typically modeled time discrete and event-discrete modeling is used mainly in event-driven domains, like traffic, transport and logistics systems.

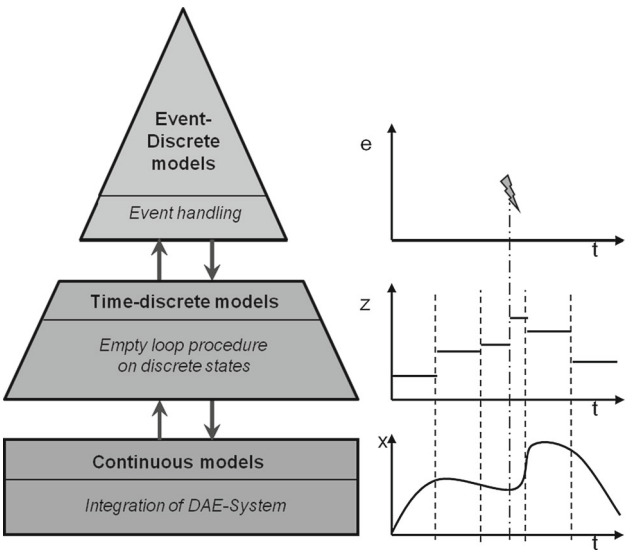

Fig. 11 Three different models and their relation

When implementing time discrete behavior, a component defines its states and realizes their calculation in the function calls provided by the meta-model. The internal states are calculated mostly out of the input signals of the components and afterwards the output signals of components are set. The simulator stops the continuous integration every preconfigured sample time point (mostly equidistantly) and starts the algorithm for the discrete system—realized by the following empty loop procedure: All discrete components are evaluated in a random order to avoid deadlocks because of periodic cycles. In doing so for every component, the internal states and output values are calculated according to the actual input values. This procedure will be repeated as long as any internal state of the whole system changes. If the fixed point is reached within a prescribed time, the continuous simulation will proceed to calculate till the next sample time point. Otherwise, the system does not converge mostly due to illegitimate loops by wrongly connected components.

In the case of event-discrete behavior, the time point at which the continuous integration is stopped and the states are calculated is triggered by events. Every component can generate and queue events. Each event is configured with the time when it fires, the dispatcher and the recipient component of the event. The simulator contains therefore an event manager, which collects all created events, queues them in the correct order and executes each one at its specific time point.

As the CoSMOS simulator supports all three types of modeling and the combined processing, there is no need to switch to another simulator when it seems more appropriate or more efficient to model parts of the continuous systems in a time or event discrete way or vice versa.

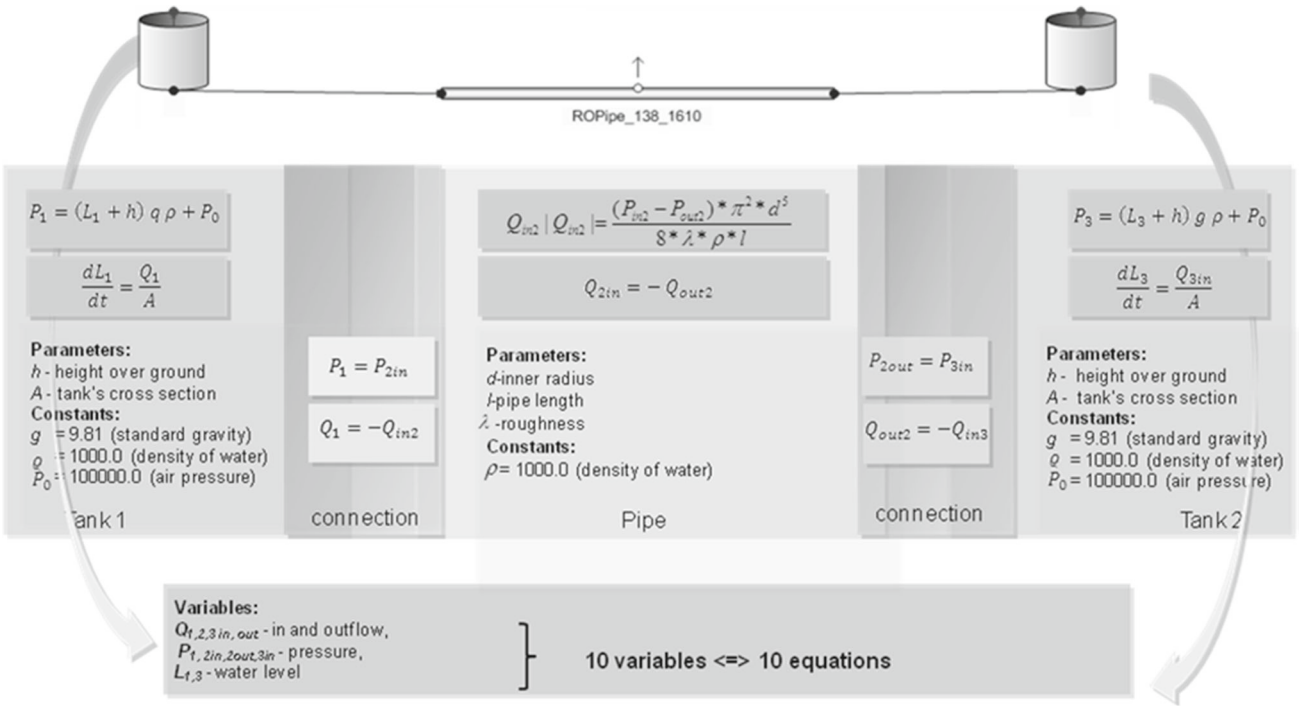

Fig. 12 Instantiated, connected components lead to the system of equations

5 Exemplification

In the following, the industrial examples of Sect. 2 will be considered regarding their implementation in the simulation environment CoSMOS as it is described in Sect. 4.

5.1 Pumping station

For the simulation of the pumping station, which is targeted by several development phases addressed in the different use cases of Sect. 3, a very flexible software system is needed: In an early conceptual phase, the latter consists of an engineering GUI, a CoSMOS simulator handling a network of components from hydraulic and automation libraries by numerical solvers for mixed discrete and continuous systems and a result evaluation. Later on, for virtual commissioning, an emulated PLC, such as SIMATIC S7 PLCSim [20], SIMATIC WinAC RTX [21] or SIMIT Emulation Platform [22], is added to the architecture to analyze real PLC software in the loop. The automation part of the system is then no longer modeled in the CoSMOS simulator, where the automation part of the model needs to be removed, but provided by SIMATIC PCS 7 OS, which is connected to the emulated PLC using standard SIMATIC technologies. Finally, for operator training, additional tools like success control and failure injection are added to the client–server architecture, see Fig. 13.

To obtain the network model of components for the simulator (CoSMOS Simulator), two approaches exist. First, the engineer could build the model manually in the engineering GUI by drag&drop, connection and individual parameterization of all components needed from the libraries. Second, less error-prone and more efficiently, the model setup can be done semi-automatically by model generation from plant engineering tools like COMOS, which contain engineering artifacts like the pipe and instrumentation diagram as well as technical data of the components like characteristic curves, geometries etc. Only additional information like initial values has then to be added by the engineer to get a complete description of the simulation model [10].

Having defined the individual settings of all clients, e.g. solver selection and tolerances for the simulator, the details of the server communication and data exchange need to be set, e.g. the length of communication time interval as well as the matching of exchange variables between different clients via matching table. These matching have to be performed for any involved client and modified if clients are added or removed.

5.2 Heat recovery steam generators (HRSGs)

Regarding the dynamic simulations of HRSGs introduced in Sect. 2.1.2, the simulator in CoSMOS incorporates dedicated libraries for boilers. The HRSG models can be assembled by simple drag-and-drop from these libraries. In order to allow for a seamless workflow, however, it is also possible to import the steady-state solution of Krawal™ [23], a stan-

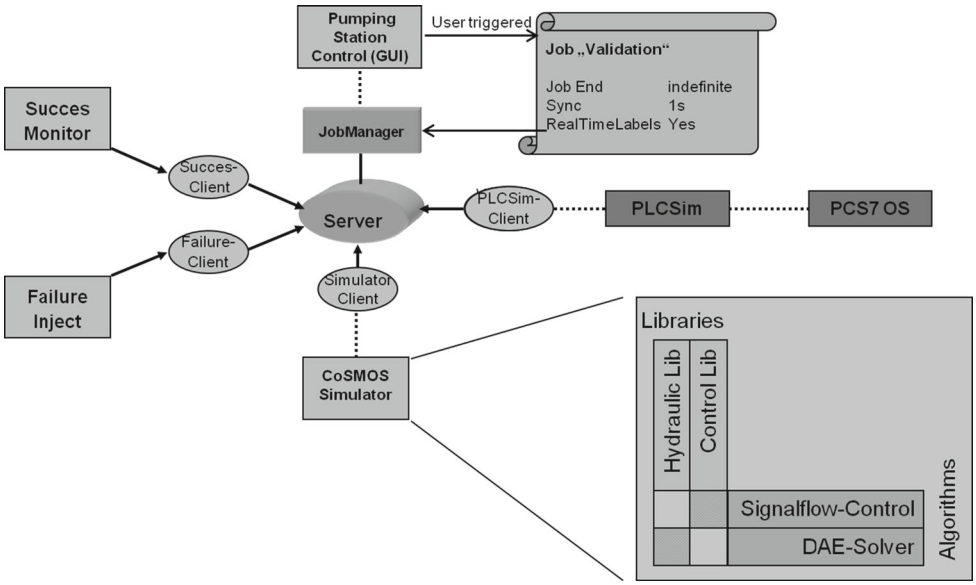

Fig. 13 CoSMOS client–server architecture for the pumping station and operator trainings. For performing a simulation, the sim-kernel is needed including the libraries for control and hydraulic. Dependent on further usage, additional clients can be added for emulated automation (e.g. PLCSim) and failure injection and success monitoring

dard Siemens tool for HRSG design, to automatically create the complete dynamic network model and fill it with parameter and initial fluid flow values. Once the HRSG models are successfully built, they pose the challenge that the underlying equation system is very large, often several thousands of differential algebraic equations are involved. In order to resolve the fluid flow behavior sufficiently, in particular for phase changes like evaporation, pipe discretizations need to be fine enough. And each cell contributes with equations for mass flow, pressure and enthalpy (or an equivalent set of state variables) plus composition. The solver in the simulator is able to deal with very large systems of equations and can switch automatically between a very efficient solving algorithm and a robust one in case of numerical difficulties.

Although some basic control and fluid cycle models can be directly built using the available libraries as well, it is often required to integrate external submodels from third party tools. To allow for such an interface, the import according to the open standard description Functional Mockup Interface for Model Exchange [14] is implemented. One overall equation system is set up in this case and solved within the simulator. Furthermore, it is possible to export the complete simulator according to the description Functional Mockup Interface for co-simulation [14]. It can then be used in other, external co-simulation environments. In contrast to model integration, each tool, including its models and specific solvers, remains independent and communication between them is only done at certain specified intervals in time. Taking a look beyond the simulator branch in Fig. 7, it becomes clear that CoSMOS itself is also such a co-simulation environment and allows for the import according to Functional Mockup Interface description for co-Simulation (FMI Client).

5.3 Sewage water management

Recalling the requirements for a simulation-based sewage management system from Sect. 2.1.3, CoSMOS provides all the necessary functionality. Some, like the continuous usage over the different lifecycle utilizations and the integration into engineering workflows, already have been exemplified by the pumping station and the HRSG example. The focus now will be on the specific requirements of the sewage management system during the usage in the operational support.

As stated above, lots of different tasks can be executed during network operation. Those tasks are completely different in the time configuration when and how they are triggered and in their internal processing. The interconnection of the simulation-based evaluations to the control center with its data management (e.g. flow values, filling levels, valve positions, weather data) requires specific algorithmic data processing during each simulation and/or optimization run. This means multiple calculation modules take part in a single evaluation and have to be orchestrated in a schedule. The data that has to be exchanged between the different modules has to be matched because the signal names, unities and magnitudes may be different.

Every task that can be performed is therefore predefined by a prepared job configuration, which in case of an execution is customized by few remaining settings and loaded by the job management. The predefinition of the job configurations defines already most of the time behavior, clients involved and the scheduling (if any) of the clients for the specific tasks.

As an example, two main tasks of a CoSMOS-based sewage management are outlined: First of all there has to be a network monitoring that runs indefinitely and estimates in a short cycle, e.g. every 15 minutes, the complete network

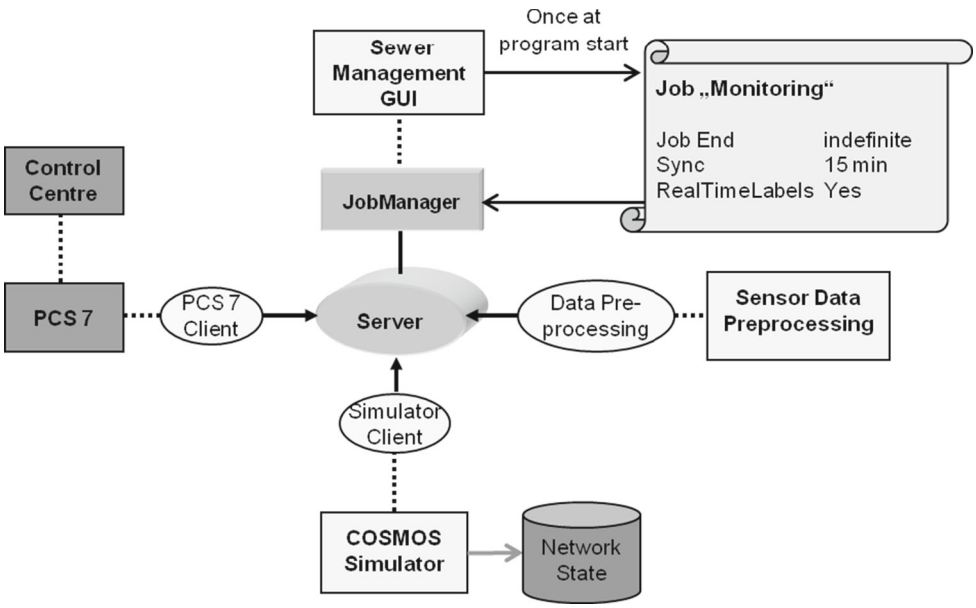

Fig. 14 CoSMOS-setup for network monitoring

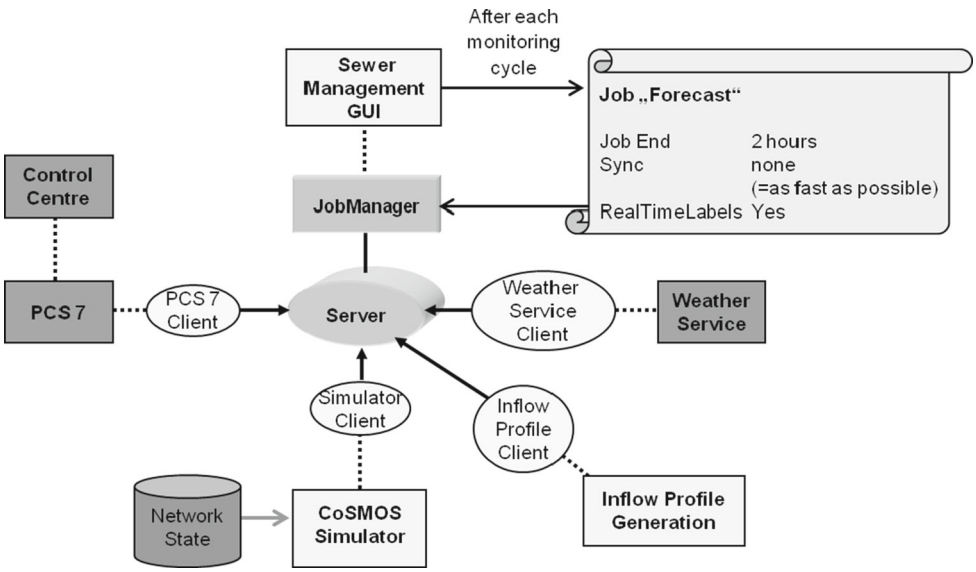

Fig. 15 CoSMOS-setup for forecast

state by taking present sensor data from the PCS 7 automation and a simulation run into account. The according CoSMOS setup is drawn in Fig. 14.

The PCS 7 and the simulation client are general clients for PCS 7 and the CoSMOS simulator. The data processing module is processing the flow and filling levels read from the PCS 7 system. After each successful calculation, the complete network state is saved in a simulator state file for further use in other tasks, like the forecast, and characteristic results are written back to the PCS 7 for display in the control center. Regarding client scheduling, the PCS 7 client has to be called before the data processing and then the simulator.

A second main task in the sewer management is the forecast of the network state in the near future, e.g. the next two hours. As this depends heavily on rain events, a weather service has to be integrated. The forecast job is automatically called by the job management after each monitoring run, takes its result as the initial state of the network and runs in parallel to the indefinite running monitoring. The CoSMOS setup now looks different (see Fig. 15).

The PCS 7 and the simulation client are in general the same even though their tasks- and therefore their configurations—have slightly changed: the simulator now loads before simulating the calculated state from the monitoring job. The PCS 7 client reads the actual control settings of e.g. control valves instead of the actual flow and filling data. A client for the general exchange with a weather forecast service and one that transforms rain predictions into network inflow profiles have been added to the setup. The schedule now is weather service,

then inflow generation, then simulator and at last PCS 7 for display.

Furthermore, when an operator wants to see the outcome of changed control settings, he—now manually—triggers the same forecast job as above after he changed the control settings in a separate operator panel.

6 Summary and outlook

The requirements on system simulations in the industrial context, from user and from developer perspective, cannot be fully met by currently available commercial tools. The challenges of phase- and task-specific setups, partly with non-expert users, a maximum in model and environment consistency and nonetheless an easy to maintain and extend software architecture demand modular, very flexible architecture concepts guaranteeing high-performance all the same.

In the presented structured approach, the challenges have been summarized in representative use cases. Subsequently, technical requirements on such an architecture concept have been deduced. The eventually presented CoSMOS concept is designed to face the challenges and use cases by implementing the technical requirements. All three industrial examples have been successfully realized in CoSMOS solutions basing on the same CoSMOS architecture.

Looking ahead, the co-simulation techniques and standards, like FMI, will evolve further, resulting in the need for more advanced sequence control algorithms and implementation of new interfaces. To get the emerging variety of co-simulation tasks under control is one of the upcoming goals of the next development steps.

Using distributed systems for a lot of common tasks is also one of the major trends in IT development. Thus, demanding distributed simulation for the simulation user, especially for the non-experts, is just obvious. The orchestration of distributed simulation applications requires various techniques and algorithms on all levels of the software. For example, web-based simulation demands for a simulation-specific browser front end and a web service capable to process simulation tasks to the evaluation cores. On the other hand, distributing simulation on solver level asks for exploitation of specific call sequences and the structure of the equation system.

Another trend when performing industrial system simulations is the extensive usage in concept and engineering studies. As the formulation and evaluation as mathematical optimization problem is often not really understood, too complex or even impossible, multiple simulation runs are executed instead. The difference in the simulation runs range from different boundary conditions to various sets of parameters and up to changes in the system topology. This procedure is not only very time consuming but also quite challenging in keeping track of the outcomes and drawing the right conclusions. A software environment that would support the user and deal automatically with some of those tasks would be mostly welcome.

This is just to name some of the main future challenges to come, when facilitating future industrial system development and operation by use of simulations.

References

1. Kim, J.-H., Lee, K., Tanaka, S., Park, S.-H. (eds.): Advanced methods, techniques, and applications in modeling and simulation. In: Proceedings Asia Simulation Conference, Seoul, Korea, Nov. 2011, Springer (2012)
2. Hartmann, D., Mahler, M.: Integration of complex 3d-simulations within systems simulations using response surfaces. In: Proceedings of the NAFEMS Seminar Strömungsberechnungen (CFD) in der Systemsimulation (2013)
3. Loper, M.L.: Modeling and Simulation in the Systems Engineering Life Cycle. Springer, Heidelberg (2014)
4. Matlab Simulink: http://www.mathworks.com/products/simulink/. Accessed 7 Nov 2014
5. AMESim: http://www.plm.automation.siemens.com/de_de/products/lms/imagine-lab/amesim/platform/. Accessed 7 Nov 2014
6. Dymola: http://www.3ds.com/products-services/catia/capabilities*/systems-engineering/modelica-systems-simulation/dymola (last access 07. Nov 2014)
7. NASA roadmaps: TA11 TA11 Modeling, Simulation, and Information Technology and Processing, NASA Space Technology Roadmaps and Priorities 2012. www.nasa.gov/offices/oct/home/roadmaps/index.html. Accessed 7 Nov 2014
8. Follmer, M., Hehenberger, P., Punz, S., Rosen, R., Zeman, K.: Approach for the creation of mechatronic system models. In: Culley, S.J., McAloone, B.J., Howard, T.J., Lindemann, U. (eds.) Proceedings of the 18th International Conference on Engineering Design ICED11, vol. 4, pp. 258–267. The Design Society, Kopenhagen (2011)
9. Pohl, K., Hönninger, H., Achatz, R., Broy, M.: Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology. Springer, Heidelberg (2012)
10. Brandstetter, V., Wehrstedt, J. C., Rosen, R., Pirsing, A.: Simulationsgestützte Entwicklung von Automatisierungssoftware. In: ATP-Edition, Fachmagazin für Automatisierungstechnische Praxis 06/2013
11. Siemens AG, Power Plants, BENSON Once-Through Heat Recovery Steam Generator. Order No. A96001–S90-A496-X-4A00, 2006, Erlangen
12. Siemens Pictures of the future (2005) http://www.siemens.com/innovation/en/publikationen/publications_pof/pof_spring_2005/elements_of_life/simulated_water_networks.htm. Accessed 11 Nov 2014
13. Pirsing, A., Rosen, R., Obst, B., Montrone, F.: Einsatz mathematischer Optimierungsverfahren bei der Abflusssteuerung in Kanalnetzen. In: GWF Wasser Abwasser, Jg.: 147, Nr. 5, Seite 376–383 (2006)

14. Modelica Association: The functional mock-up interface, https://www.fmi-standard.org/. Accessed 7 Nov 2014
15. Pumhössel, T., Hehenberger, P., Boschert, S.: On the advantages of using reduced system models in the model-based development of mechatronic systems. In: Proceedings of 2nd Workshop on Mechatronic Design, Paris, France (2013)
16. SIMATIC PCS7 Operator System http://w3.siemens.com/mcms/process-control-systems/en/distributed-control-system-simatic-pcs-7/simatic-pcs-7-system-components/operator-system/pages/operator-system.aspx. Accessed 7 Nov 2014
17. TISC: http://www.tlk-thermo.com/de/softwareprodukte/tisc.html. Accessed 7 Nov 2014
18. ICOS: http://www.v2c2.at/icos/. Accessed 7 Nov 2014
19. Günther, M., Rentrop, P., Feldmann, U.: CHORAL - a one step method as numerical low pass filter in electrical network analysis. In: Scientific computing in electrical engineering. In: Proceedings of the 3rd International Workshop, 20–23 August 2000, Warnemünde, Germany. Hrsg.: Rienen, U. van, Günther, M., Hecht, D. Springer, Berlin, 199–215 (2001)
20. SIMATIC S7 PLCSim: http://w3.siemens.com/mcms/simatic-controller-software/de/step7/simatic-s7-plcsim/seiten/default.aspx. Accessed 7 Nov 2014
21. SIMATIC WINAC RTX: http://w3.siemens.com/mcms/programmable-logic-controller/de/software-plc/simatic-winac-rtx/seiten/default.aspx. Accessed 7 Nov 2014
22. SIMIT Emulation Platform: http://www.industry.siemens.com/services/global/de/it4industry/produkte/simulation/simit-emulation-platform/seiten/technische-daten.aspx. Accessed 7 Nov 2014
23. Altpeter, R., Schiller, N., Weissenberger, B., Schweizer, R: Praktische Erfahrungen mit Online-Diagnosesystemen unter Einsatz des Kreislaufprogramms KRAWAL-modular, VDI Hannover 03.06.03